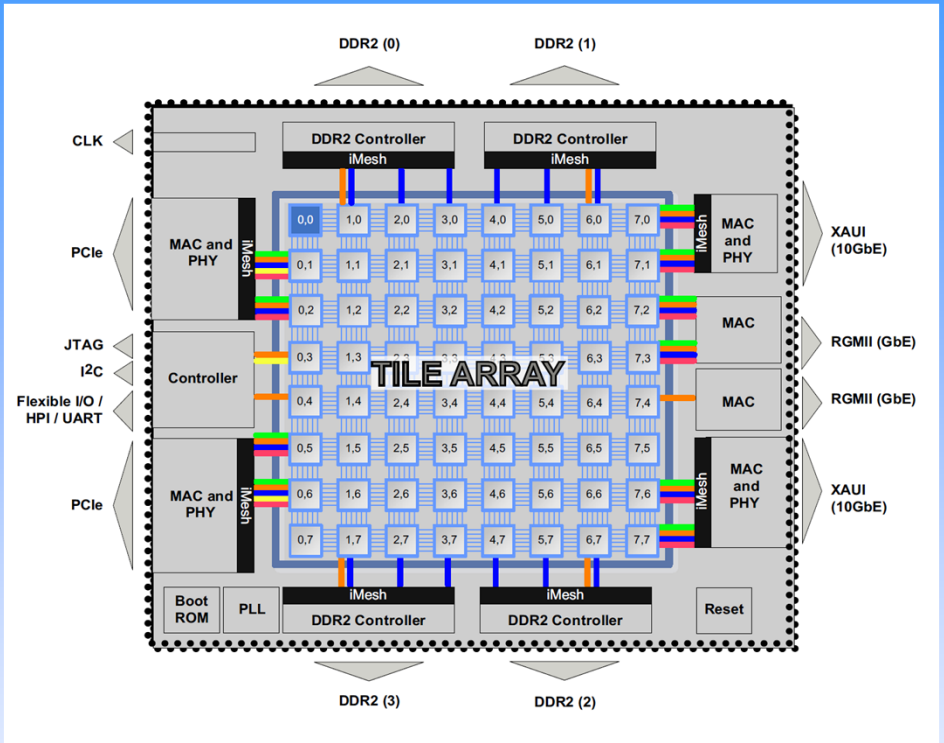
Reliable Multicore Processors for NASA Space Missions

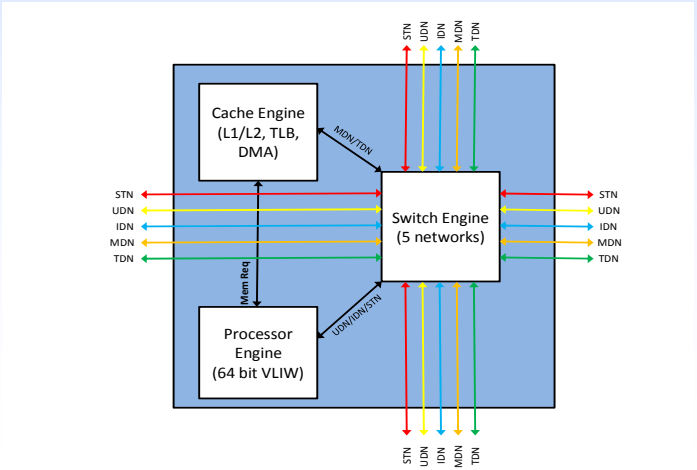
Carlos Villalpando, David Rennels, Raphael Some
carlos@jpl.nasa.gov, rennels@cs.ucla.edu, rsome@jpl.nasa.gov
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr. Pasadena, CA 91109

Manuel Cabanas-Holmen
manuel.f.cabanas-holmen@boeing.com
The Boeing Company
PO Box 3707, MS 42-57, Seattle, WA 98124

This work was done under the High Performance Processing task of the Advanced Avionics and Processor Systems Project, which is a part of NASA's Exploration Technology Development Program
(c) 2011 California Institute of Technology. Government sponsorship acknowledged.

Tilera TILE64 Architecture Base



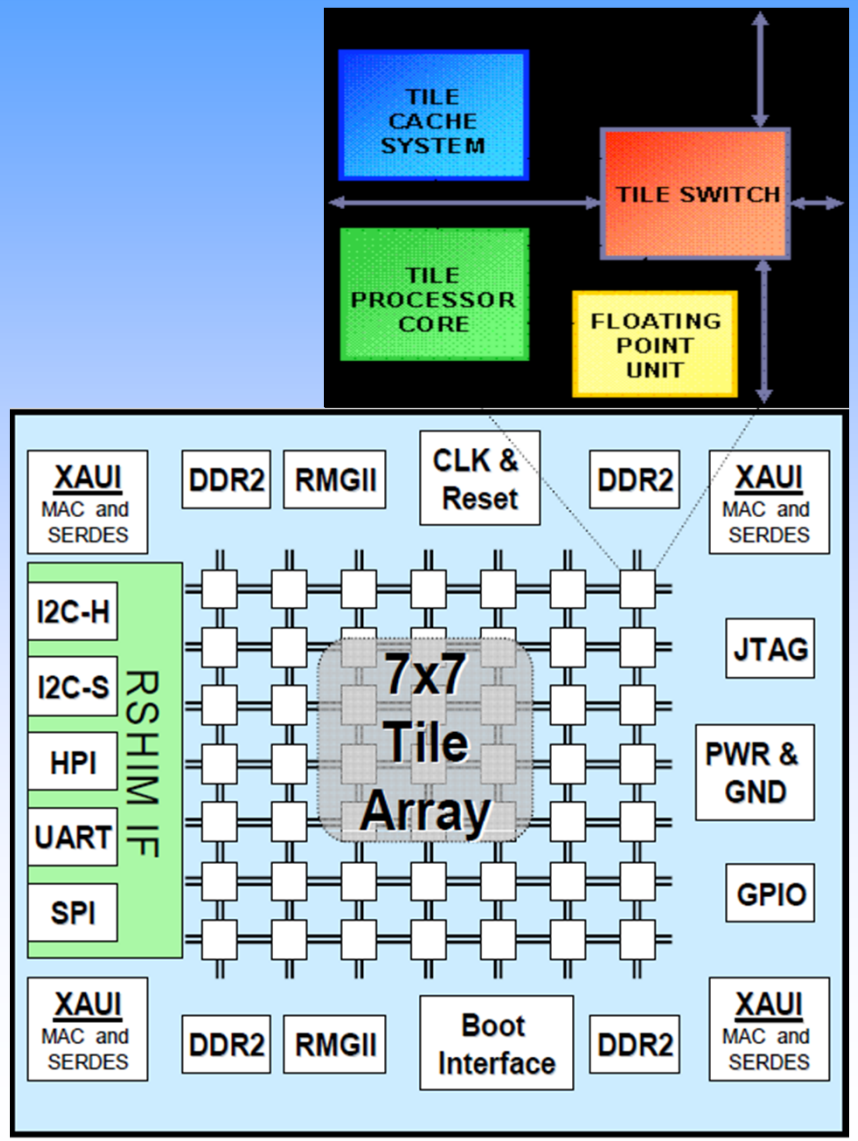


- 8x8 Array of Processing Elements (PE)
- 5 data meshes in a grid
- Each PE is a 3 instruction VLIW architecture
- Each PE contains its own TLB
- 8K L1 Data, 8K L1 Instruction and 64K L2 unified cache per Processing Element.
- 4 DDR2 Controllers
- Multiple high speed Ethernet/XAUI controllers.

TILE64 Fault Tolerance Challenges

- Original TILE64 designed for terrestrial applications.
- Many parallel networks, but no redundancy. Each network performs a different function.
- No automatic re-routing of failed network nodes. Fixed dimension-ordered routing scheme does not avoid failed nodes
- No error correction on memories or mesh networks. Bit errors may propagate or hang network routers
- SEU Soft logic cells and memories
- Few fault tolerance features. Only parity on caches, processor is not fault tolerant, etc.

OPERA Maestro Modifications



- Logic Cells Replaced with Boeing
- 90nm Radiation Hardened by Design (RHBD) Libraries
- Array reduced to 7x7 grid.
- Aurora FPU added to each PE as a co-processor
- PCIe replaced by additional XAUI ports.
- I/O ports replaced by Rad-tolerant designs
- On-chip memories replaced by Artisan standard cells, with bit interleaving and EDAC
- SET temporal filters added to clock and data and scan inputs.
- DDR controllers contain both DDR1 and DDR2 functionality, with BIST and loopback testing

Potential Space Use Cases

Payload Processing

- Low reliability requirements
- Low availability requirements
- High throughput requirements
- Non real-time, no Command & Data Handling (C&DH) responsibility
- Long lifetime requirements
- Detect and report any uncorrected errors

Navigation Processing

- Higher reliability requirements
- Low availability requirements
- High throughput requirements
- Real-time, no C&DH
- Variable lifetime requirements
- Tighter error handling requirements: i.e. recover and work through faults

Robotic C&DH

- High reliability requirements
- High availability requirements, depending on system design
- Low throughput requirements
- Hard real-time
- Long lifetime requirements
- Tighter error handling requirements: i.e. recover and work through faults

Human/Life Critical Systems

- High reliability requirements
- High availability requirements, depending on system design
- Low throughput requirements
- High fault tolerance requirements
- Single Processor insufficient. System level redundancy required to avoid single-points-of-failure

Error Detection and Recovery

- Although design techniques such as RHBD reduce the error rate, it does not eliminate it. Error detection and recovery is still needed to deal with any detrimental errors that may still occur
- Software Implemented Fault Tolerance (SIFT) is a tool, but needs architecture support. For example, a SIFT algorithm may protect data integrity, but can be defeated by an upset on the program counter or program memory.
- Gross Error Detection and Recovery:
  - Monitor process or devices that detect outliers/crashes which will roll in hot-spares and/or reboot the processor
- Comprehensive Error Detection and Recovery
  - Algorithm Based Fault Tolerance—built-in redundancy in computation
  - Checkpoint-and-rollback—go back to a “known-good” state when errors detected
  - Triplication of processors/software modules with voting

Architecture Support for Reliability

- **Reliable messaging network issues**
  - Message routing network must be tolerant to errors in message routing information. Error may prevent a packet from being delivered, but it must not place network in unusable state
  - Message must be delivered, or reported that it did not get delivered.
- **Reliable Messaging network techniques**
  - On-chip messaging network should include error checking and/or correction. Checksums and/or SEC/DED.
  - Network should include support for determining if a message made it to its intended destination, either by hardware support, or overlying redundant software support. Place both source and destination info in packet so that message misdelivery can be properly reported.

Architecture Support for Reliability (cont.)

- Reliable messaging network (cont.)
  - Software techniques can address many Processor to Processor message reliability techniques, but Processor to I/O device message reliability requires hardware support
  - Allow autonomous re-routing around failed routing elements. Fixed routing schemes limit the ability to avoid failed elements.
  - Propagation limits. Limit the ability for messages to propagate beyond their intended regions. Hardware support for preventing a message from crossing a physical boundary.

Architecture Support for Reliability (cont.)

- Reliable memory system
  - Reliable caches. SEU resistant memory cells. SEC/DED on memories to correct errors and report any uncorrectable errors.
  - Memory protection. Processes and processors must be prohibited from accessing memory they do not have privileges to. MMUs, hardwalls, etc.
  - Error Detection/Correction on external memories

Software Support for Fault tolerance

- **Triplication in software.**
  - Run 3 copies of the software in different regions of the processor. Hardware can support this by limiting message propagation preventing errors from spilling into the other 2 copies
- **Algorithm Based Fault Tolerance**
  - Place redundancy in the actual algorithm with the goal of self-consistency checking
- **Checkpoint/Rollback**
  - Regularly checkpoint software state and return to a known good state when the hardware reports an uncorrected error. Hardware can support this with accelerated context/program state save and restore, potentially into hardened memory.

What methods to use?

- Depends on use case and reliability requirements of each use case.
  - Analyze the effectiveness of each method against the required performance
  - Some requirements may require redundant hardware and software to meet the error rate requirements, some may be tolerant of re-starting.

Conclusions

- Industry has provided a good baseline for reliable multicore processors for space applications.
- Projects/Missions must now compare their requirements with what is available and work towards expanding what is available.